INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Tag-based Fair Bandwidth Sharing for Responsive and Unresponsive flows

Antoine Clerget, Walid Dabbous

N° 3846

December 1999

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--3846--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Tag-based Fair Bandwidth Sharing for Responsive and Unresponsive flows

Antoine Clerget, Walid Dabbous

Thème 1 — Réseaux et systèmes
Projet Rodéo

Rapport de recherche n° 3846 — Décembre 1999 — 20 pages

Abstract: Finding an appropriate end to end congestion control scheme for each type of flow, such as real-time or multicast flows, may be difficult. But it becomes even more complex to have these schemes be friendly among themselves and with TCP. The assistance of routers within the network for fair bandwidth sharing among the flows is therefore helpful. However, most of the existing mechanisms that provide this fair sharing imply complex buffer management and maintaining flow state in the routers. In this paper, we propose to realize this fair bandwidth sharing *without per-flow state* in the routers, using only a trivial queueing discipline. Packets are tagged near the source, depending on the nature of the flow. In the core of the network, routers use FIFO queues, and simply drop the packet with the highest tag value in case of congestion. We call our scheme $\mathcal{T}$UF , Tag-based Fair Queueing for Responsive and Unresponsive Flows, and compare it with the Core Stateless Fair Queueing (CSFQ) proposed recently by Stoica et al [6]. We believe our approach is simpler, avoids in particular parameter tuning and label rewriting in the core routers, while providing similar performances. Moreover, tagging flows depending on their responsiveness to losses, as proposed in $\mathcal{T}$UF , avoids poor performances in bursty environments, as in CSFQ.

Key-words: Fair Queueing, congestion control, max-min fairness, router filters, multi-rate sessions, responsiveness, Tags

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Utilisation de label pour un partage équitable de bande passante

Résumé : Définir des mécanismes de contrôle de congestion de bout en bout pour des flôts avec contraintes temporelles, ou des flots multipoints peut être assez complexe. Ceci l'est d'autant plus qu'il faut veiller à adopter un comportement compatible avec celui des flots TCP. Obtenir l'aide des routeurs dans le réseau pour partager équitablement la bande passante est donc utile. Les propositions existantes utilisent des mécanismes complexes de gestion de files d'attente et maintiennent un état par flot dans les routeurs. Nous proposons ici de réaliser ce partage équitable sans introduire d'état dans les routeurs au coeur du réseau. Les paquets sont marqués près de la source, en tenant compte de la nature du flôt. Dans le coeur du réseau, les routeurs se débarrassent des paquets marqués avec les plus hautes valeurs en cas de congestion. Nous comparons notre protocole, TUF, Tag Based Fair Queueing for Responsive and Unresponsive Flows, avec celui proposé récemment par Stoica et al., CSFQ (Core Stateless Fair Queueing). Nous obtenons avec notre approche des performances semblables. Toutefois, en tenant compte de la nature des flots, notre mécanisme est plus robuste lorsque le traffic externe est très irrégulier.

Mots-clés : Equité, contrôle de congestion, filtres, protocoles, Internet, labels, multidébit

1 Introduction

With traditional end to end congestion control schemes, sources adapt their rate in response to congestion signals from the network. As in TCP, these signals often correspond to packet losses in the congested routers. Fairness is achieved by the congestion control algorithm in the end hosts : sources tend to send at the same rate because they undergo the same loss rate. However, leaving all the responsibility of congestion control to end hosts is sometimes not possible. This is for example the case for multicast flows, where receivers heterogeneity makes rate adaptation non-trivial for the source. Moreover, having different schemes cooperate makes it even more complex. Fair bandwidth sharing in the routers can therefore be useful, and can protect well-behaved flows from ill-behaved ones.

To provide fair bandwidth allocation between a responsive TCP flow and an agressive - unresponsive flow in a congested router, we need to differentiate their loss rates. Using techniques maintaining flow state in the routers at the core of the network increases the implementation complexity. This complexity is often an obstacle to their massive deployment. We have designed our congestion control scheme $\mathcal{T}$UF to be very simple and to avoid per-flow state in the routers. As no state is present in the routers, state has to be present in the packets to enable distinct behaviors (i.e. loss probability) between the flows. That state is a "$\mathcal{T}ag$", a numeric value carried in one of the packet's field. The internal state of the congested router (e.g., the average queue size) and the tags of the packets present in its queue are the only elements used by the router to make a drop decision.

The paper is structured as follows : In section 2, we discuss previous work related to stateless fair queueing. Section 3 starts by presenting our objectives and a basic overview of $\mathcal{T}$UF . We then go into further details and implementation issues. Section 4 describes some of the strengths of $\mathcal{T}$UF . Section 5 presents simulation results and section 6 concludes the paper.

2 Related Work

Fair bandwidth allocation techniques have been studied to protect well-behaved flows from ill-behaved ones. Until recently, these techniques classified packets into flows, updating variables pertaining to that flow. They then either maintained a separate queue for each flow (*per-flow queueing*), or used one queue with flow state to determine the packets to drop in case of congestion (*per-flow dropping*) as in FRED [1]. Among the variants of Fair Queueing [2] [3] [4], that fall in the first category, Deficit Round Robin [5] achieves nearly perfect fairness with limited complexity.

However, maintaining that flow state has a cost and raises scalability issues in high speed networks that can jeopardize the massive deployment of these techniques. Recently, Stoica et al. proposed *Core*-Stateless Fair Queueing (CSFQ) [6], in order to approximate fair queueing while limiting flow state to the edge of network and removing it from core routers. Cao et al. proposed Rainbow Fair Queueing (RFQ) [7], a similar scheme that avoids fair share rate calculation in the core routers and that is better adapted to layered encoding applications.

A previous version of $\mathcal{T}$UF [8] was conceived originally as an alternative to Receiver-driven Layered Multicast [9] for multicast congestion control in heterogenous environments. As with RFQ, layered encoding applications can benefit from this scheme.

The fair bandwidth sharing mechanism $\mathcal{T}$UF proposed here differs from those mentionned above in the following way :

- Contrarily to CSFQ, the $\mathcal{T}$UF tag does not represent the rate of the flow. It keeps its value during all the packet's lifetime. We therefore avoid label rewriting in each router.
- $\mathcal{T}$UF routers consist in a very simple queueing discipline. Contrarily to previous work, we do not need to estimate fair share rates (CSFQ) or color thresholds (RFQ). We avoid having to set parameters that impact on the router's reactivity ($update_{int}$ for RFQ and K_c, K_α, for CSFQ).
- State is introduced in the packets depending on the nature of the flow and its responsiveness to losses. Thus, $\mathcal{T}$UF is better adapted to responsive flows in bursty environments.
- $\mathcal{T}$UF provides a smooth distribution of packet losses.

3 TUF : Tag-based Fair Queuing for Responsive and Unresponsive Flows

3.1 Objectives

3.1.1 Max-min Fairness

Our primary objective is to allocate network resources fairly among the flows. We here focus on the bandwidth allocation of a router in the core of the network. Our fairness objective can be summarized as follows : *we say that we allocate bandwidth fairly if it is not possible to increase the satisfaction of a flow without simultaneously causing the decrease in the satisfaction of a more constrained flow.* This qualitative property, used to define fairness, is called **max-min fairness** [10].

Let's suppose that we have n flows, $F_1, \ldots, F_n$. Let $\hat{R}_i$ be the rate of flow F_i upstream (arriving at the router), and R_i be the rate of the flow downstream (leaving the router). The bandwidth allocation is fair if $\forall i, R_i = min(\hat{R}_i, R_{fair})$. R_{fair} is the fair share rate at the router. It corresponds to the maximum bandwidth a flow should get. R_{fair} is set to fill the router's capacity C ; it is the unique solution to :

$$C = \sum_i R_i = \sum_i min(\hat{R}_i, R_{fair})$$

A flow F_i is either :

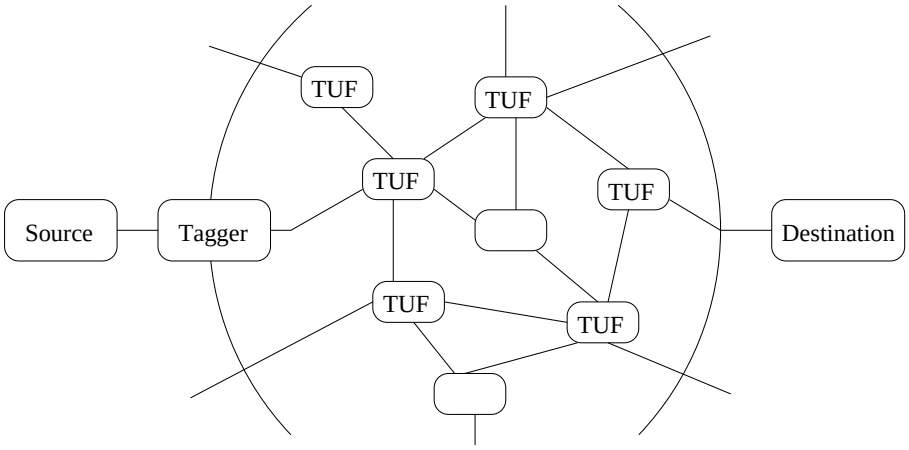

Figure 1: Our tagging architecture

- Constrained : the bandwidth obtained is, as all constrained flows, equal to the fair share rate, and is larger than that obtained by unconstrained flows. $\forall j, F_j$ constrained, $R_i = R_j = R_{fair}$
- Unconstrained : the bandwidth requested by the flow is less than the fair share rate, that it could get. The router therefore forwards all its packets and the loss rate is 0. $\forall j, F_j$ constrained, $R_i = \hat{R}_i \leq R_j = R_{fair}$

Weighted fairness

To generalize our fairness criteria, we can define relative priorities by associating weights with flows. A flow with a weight of $w = 2$ can get twice as much bandwidth as a flow with a weight of 1. The definition above does not change, except that the rates R_i and $\hat{R}_i$ are replaced by R_i/w_i and $\hat{R}_i/w_i$. Thus, $\forall i, R_i/w_i = min(\hat{R}_i/w_i, R_{fair})$.

3.1.2 Stateless congestion control

A second objective is to avoid per-flow state in the high-speed backbone routers and to maintain our algorithms scalable and easily implementable. A tagging entity, called the "tagger", is responsible for placing a tag in each packet. The tagger is the only entity maintaining flow state, necessary for tagging. This is why the tagger should be kept close to the source. Tagging is done either by a router at the edge of the network, or ideally by the source itself. Figure 1 presents our global architecture.

Not having flow state in the routers that are in the core of the network has two consequences :

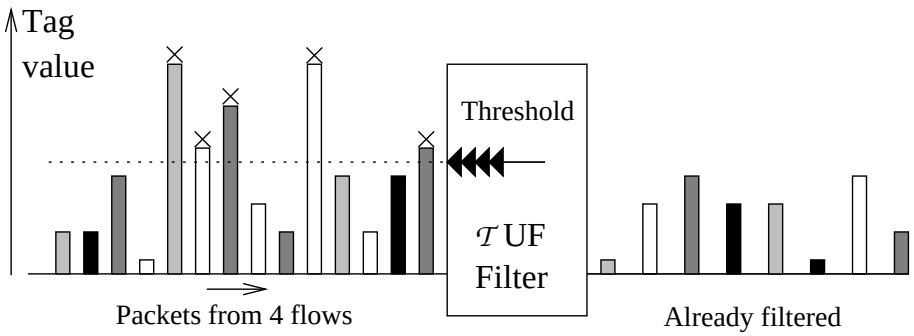

Figure 2: A $\mathcal{T}$UF router, modeled as a simple filter

- Tagging should not depend on the state of the network. We do not want specific feedback from the routers in the network (or the destinations).
- Dropping decisions in the core routers are not based on per-flow information, but on the tags of the packets present in the queue.

3.2 $\mathcal{T}$UF overview

This section presents $\mathcal{T}$UF , our Tag-based Fair Queueing scheme. The $\mathcal{T}ag$ is a numeric value carried in a field, in the packet header. It is used by the routers to decide if they should enqueue or drop a packet. We chose to give high values to the tags of the packets that will be more eligible for discard, and low values to the tags of packets that are less eligible for discard. The tag therefore represents a **drop precedence**. A $\mathcal{T}$UF router forwards the packets with low values and discards those with high values in case of congestion. We simply model our routers as bufferless filters that let through the packets with tags lower than threshold K as shown on figure 2.

As the load of the router increases, more packets need to be dropped, and the router reduces its tag threshold K. On the contrary, when the load decreases, the router increases its tag threshold, maintaining a full utilization of its capacity. The threshold reaches the maximum value K_{max} when the router forwards all the packets. In section 3.2.2, we will see that we can realize a good approximation of this filter with a very simple queuing algorithm : when a packet arrives and the queue is full, the router drops from its queue the packet with the highest tag.

Let us consider a greedy source that sends packets through router A. The tag threshold for router A is K, and the flow receives the fair share rate R_{fair}. Let us suppose that the packets are now rerouted towards a router B that has the same tag threshold K. The same packets will be filtered out, and the flow will therefore achieve the same fair share rate R_{fair}. A threshold K is therefore representative of a fair share rate R_{fair}. We can "convert" rates to integer tags using an increasing function T, a constant of the protocol : $\mathcal{T}ag = T(Rate)$.

Section 3.3.1 explains how we choose the function T. Intuitively, when a source or tagger sends a packet with the tag $\mathcal{T}ag$, it asks each router on the path to *"forward the packet if the fair share rate is greater than $T^{-1}(\mathcal{T}ag)$"*. We will now describe the tagging algorithm at the tagger and the queueing discipline in the core routers used to achieve fairness.

3.2.1 The tagging algorithm

This section describes how the tagger sets tags in the packets in order to achieve fairness in a congested router on the path. Let us consider n flows $F_1, \ldots F_n$. For generality, we define the flows' weights $w_1, \ldots, w_n$. We suppose that we can model these sources, and that we know the throughputs achieved by these flows when they are submitted to a loss rate p : $B_1(p), \ldots, B_n(p)$. The functions B_i are decreasing functions, since the throughputs achieved decrease when the loss rates increase. Sections 3.4.1 and 3.4.2 describe how these functions are defined in practice.

Weighted fairness is achieved if, for all tag thresholds K:

- For constrained flows : $R_i/w_i = B_i(p_i)/w_i = T^{-1}(K) = R_{fair}$
- For unconstrained flows : $R_i/w_i = B_i(p_i)/w_i \leq T^{-1}(K) = R_{fair}$ and $p_i = 0$

where p_i is the loss rate for flow i when the threshold is K.

Theorem 1 *A "fair" tagging function for packets of flow F_i is $T(B_i(X)/w_i)$, where X is a stochastic variable, uniformly distributed over the interval [0,1].*

PROOF :
For constrained flows :

$$\begin{aligned} B_i(p_i)/w_i &= B_i(Probability(\mathcal{T}ag > K))/w_i \\ &= = B_i(Probability(T(B_i(X)/w_i) > K))/w_i \\ &= B_i(Probability(X < B_i^{-1}(w_i.T^{-1}(K)))/w_i \\ &= B_i(B_i^{-1}(w_i.T^{-1}(K)))/w_i = T^{-1}(K) = R_{fair} \end{aligned}$$

For unconstrained flows :

$$\begin{aligned} p_i = 0 &\Rightarrow Probability(T(B_i(X)/w_i) > K)) = 0 \\ &\Rightarrow Probability(B_i(X)/w_i > T^{-1}(K)) = 0 \\ &\Rightarrow B_i(0)/w_i \leq T^{-1}(K) = R_{fair} \end{aligned}$$

To process a packet, the tagger determines its flow, the corresponding function B and weight w. The tagger then determines a value for X, uniformly distributed between 0 and 1, computes $T(B(X)/w)$ and sets the tag value in the corresponding packet field.

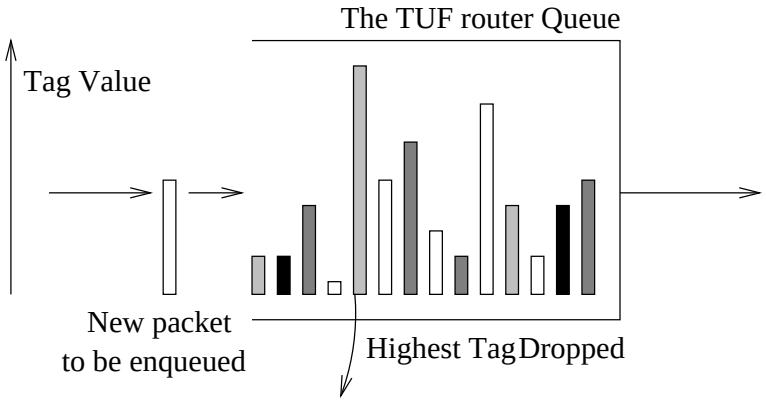

Figure 3: The queueing discipline in a $\mathcal{T}$UF router

3.2.2 The drop algorithm

The drop mechanism in our $\mathcal{T}$UF routers has been designed to be very simple and easily implementable. The queuing discipline adopted is shown on figure 3. When the queue is not full, our router forwards the packet like a standard FIFO router. In case of congestion, when the queue is full, the packet with the highest tag is dropped.

We simulated this queueing discipline and compared it with the bufferless filter model presented in section 3.2. The results shown on figure 4 show that they have a similar behavior. For the simulation, our tags were uniformly distributed between 0 and 2047. Our queueing discipline was simulated for the M/D/1/64 queue (Poisson arrival rate is 2000 packets/s, simulation time is 1000s, service time is 1ms).

The $\mathcal{T}$UF router actually does not actually "set" or "adapt" its threshold K to its congestion state. The 'equivalent' threshold in the bufferless filter model depends on the tags of the packets in its queue. When the router becomes more congested, the queue fills up with low tagged packets, and the equivalent threshold in the bufferless filter model decreases.

We will try to explain intuitively why the $\mathcal{T}$UF router behaves as a filter. Let us consider the portion of the traffic with tags below K and the corresponding utilization factor[1] ρ_K. For small values of K, where $\rho_K \ll 1$, packet losses occur when the queue fills up completely with packets tagged below K. This is therefore a rare event. For high values of K, where $\rho_K \gg 1$, the capacity of the router is almost completely filled with lower tagged packets. Since packets with tags above K can not be forwarded to the detriment of lower tagged packets, they will therefore almost always be dropped. Finally :

- $LossProbability(k) \approx 0$ when $\rho_k \ll 1$
- $LossProbability(k) \approx 1$ when $\rho_k \gg 1$

[1] traditionally defined as the ratio between the average arrival rate and the average service rate

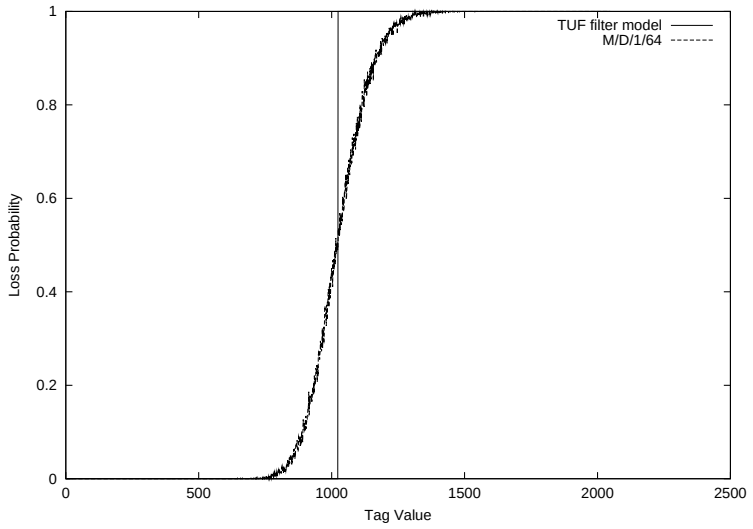

Figure 4: Comparing our queueing discipline to the TUF filter model

3.3 Optimizing $\mathcal{T}$UF

3.3.1 Defining the tag-rate conversion function

In our tagging algorithm (section 3.2.1), we used a generic tag/rate conversion function T, that we have not defined yet. This function must be an increasing function that associates rates - real values - to a limited number of tags - integer values. We should try to use a function that maps as well as possible these tag integer values $(0, \ldots, K_{max} - 1)$ to the rates we observe in the Internet. We should indeed try to minimize the error we make by rounding the ideal fair share rate to make it match one of the values $T^{-1}(0), T^{-1}(1), \ldots, T^{-1}(K_{max} - 1)$. Let's consider that the rates we observe in the Internet are bounded by R_{min} and R_{max}, for example 2^{-10} and 2^{22} packets per second. To minimize the uncertainty on the fair share rate, we choose a logarithmic scale as shown on figure 5.

$$T(r) = \left\lfloor K_{max} \cdot \frac{log(r) - log(R_{min})}{log(R_{max}) - log(R_{min})} \right\rfloor$$

Hence, for rates in $[R_{min}, R_{max}]$, the error is bounded by $\epsilon = (R_{max}/R_{min})^{1/K_{max}}$. The values R_{max} and R_{min} proposed above correspond to the rates of 1 Byte/s and 4 GBytes/s for packets of 1 KBytes. With an 11 bit field for the tag ($K_{max} = 2048$), the approximation is bounded by $\epsilon < 2^{1/32} - 1 \approx 1\%$ and the general expression of T is :

$$T(r) = \lfloor 2^6 . (log_2(r) + 10) \rfloor$$

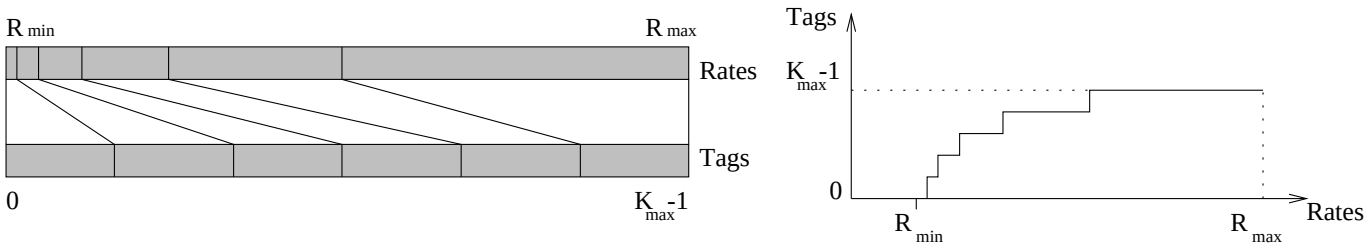

Figure 5: The tag-rate conversion function

3.3.2 Avoiding bursts of losses

The tagging function presented in section 3.2.1 uses a stochastic variable X uniformly distributed between 0 and 1. This can be a random value. However, by choosing the value of X (still uniformly distributed between 0 and 1) so as to interleave low tag values with high tag values, we can improve the loss pattern and avoid bursts of losses. Since B is a decreasing function, low tag values are obtained with values of X close to 1, and high tag values with values of X close to 0. What we propose is to choose the sequence of values X_n taken by X for a flow as follows :

The value of the variable X for packet n from flow F is $X_n = I(n + D_F)$, where D_F is a random integer, used to avoid flow synchronization, and :

$$\begin{array}{rccc} I: & \mathbb{N} & \longrightarrow & [0,1[\\ & \sum_{i=0}^{\infty} b_i.2^i & \mapsto & \sum_{i=0}^{\infty} b_i.2^{-i-1} \end{array}$$

For example, with $D_F = 1$, X would take the following values : $1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, 1/16, \ldots$, alternating high and low tags.

3.4 Tagging TCP and UDP flows

3.4.1 TCP flows

Models of TCP throughput depending on the loss rate and round trip time have been proposed in the literature. The TCP model presented in [11] is a good approximate model for TCP (b is the number of pqckets acknowledged by a received ACK, and t_0 is the timeout timer) :

$$B(p) = min\left(W_{max}/RTT, \frac{1}{RTT.\sqrt{\frac{2bp_i}{3}} + t_0.min(1, 3\sqrt{\frac{3bp_i}{8}})p_i(1+32p_i^2)}\right)$$

The tagger needs to keep track of the connection's RTT, by observing the acknowlegments. It updates, exactly as the TCP source, the value of the timeout timer based on the RTT estimations. This becomes very simple if tagging is done at the source, since the source

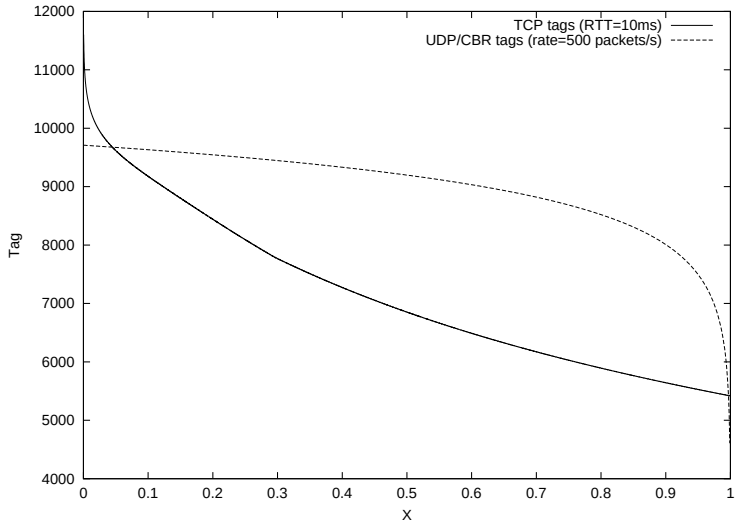

Figure 6: Tags for TCP and UDP flows

already keeps track of these parameters. Figure 6 shows the tag value as a function of X for TCP flows.

$$Tag_{tcp} = T\left(min\left(W_{max}/RTT, \frac{1}{RTT.\sqrt{\frac{2bX_n}{3}} + t_0.min(1, 3\sqrt{\frac{3bX_n}{8}})X_n(1+32X_n^2)}\right)\right)$$

3.4.2 UDP flows

Estimating the throughput as a function of the loss rate is something very simple for a CBR source whose rate R is known:

$$B_{cbr}(p) = R.(1-p)$$
$$\mathcal{T}ag_{cbr} = T(R.(1-X_n))$$

This formula can be used for unresponsive flows, that can be assimilated to CBR sources over short time periods. (For responsive flows, that use their own congestion control protocol over UDP, sources will have to set themselves the packet tag, since they alone know their behavior facing losses). The tagger must now estimate the rate R of the flow. For that purpose, it updates the rate estimation using an exponential averaging. The rationale for using this kind of averaging has been discussed in CSFQ [6].

$$R_{new} = (1 - e^{-\Delta t/K})/\Delta t + e^{-\Delta t/K}.R_{old}$$

where Δt is the packet interarrival time, and K is a constant set to $100ms$.

Figure 6 show the tag values depending on the value of X for UDP/CBR flows.

3.5 Implementation issues

3.5.1 Placing the tag in the packet header

We need to find at least 11 bits in the IP packet header for the tag. It is possible to add a new IP option. This will however significantly increase the packet's processing time. As pointed out in [12], very few packets (0.22%) are actually fragmented. A second possible implementation is to use the IP identifier field for the $\mathcal{T}ag$ when the pair (More Fragment and Fragment Offset) are both set to zero (i.e. the packet is not fragmented). Fragmented packets are ignored by $\mathcal{T}$UF and forwarded as usual. A reserved value of the TOS byte is used to indicate that the packet is transporting a $\mathcal{T}$UF tag.

3.5.2 Incremental deployment

For an incremental deployment, we must make sure that :

- Non-tagged packets can traverse $\mathcal{T}$UF routers. Minimum bandwidth should be reserved for these packets. We can use two separate queues that we serve in a weighted round-robin manner : the queue with tagged packets, and the queue with non-tagged packets.
- Tagged-packets can go though non-$\mathcal{T}$UF routers. Of course, fair bandwidth sharing is not assured in non-$\mathcal{T}$UF routers.

4 Strengths and perspectives

4.1 Multiple congested routers

We clearly understand that our mechanism is not affected by the presence of non-congested routers along the flow's path, since these do not change the tags and do not drop packets. However, we may wonder what happens when the flow crosses multiple congested routers. We modeled $\mathcal{T}$UF routers as filters that drop packets whose tag exceed a tag threshold K. Having multiple congested routers along a flow's path is equivalent to having only the bottleneck router congested, as shown on figure 7.

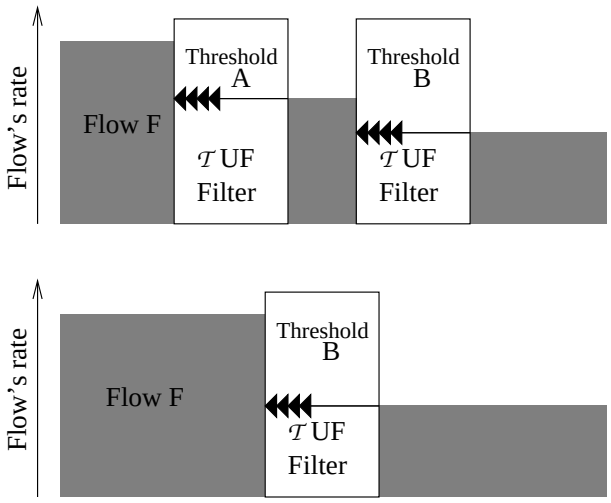

Figure 7: Multiple congested gateways

4.2 $\mathcal{T}$UF and rerouting

Our protocol is not sensitive to routing changes : $\mathcal{T}$UF routers do not contain flow state, and react immediately to the arrival of a new flow, by reducing the fair share rate. However, routing changes become a problem if they occur between the source and the tagger, since the tagger holds information concerning the flow.

4.3 $\mathcal{T}$UF and multicast

As explained in [8], such a congestion control scheme based on Fair Queueing in the routers is particularly adapted to multicast environments, since each receiver in the multicast group receives its fair share rate, regardless of the presence of slower receiver in the group.

4.4 $\mathcal{T}$UF and layered encoding

If tagging is done at the source, layered encoding applications can benefit from $\mathcal{T}$UF by asking the network to drop packets from the higher layers first. Suppose that our application sends its data into three layers, at the same rate in each layer. To send data in the first layer, the application will choose X uniformely distributed in the interval $[2/3, 1[$ and compute the tag. For the second layer, X will be uniformely distributed in the interval $[1/3, 2/3[$, and for the third layer in the interval $[0, 1/3[$. As long as the application keeps sending at an equal rate in each layer, the overall X will be uniformely distributed between 0 and 1, and we will achieve fairness. The packets dropped first will be those of the last layer (corresponding to low values of X). This can be generalized to any number of layers with various rates.

5 Simulations

In this section, we evaluate our proposal by simulations using the $ns-2$ simulator [13]. We compare the throughputs and fairness achieved using the following mechanisms in the routers :

- TUF : Our Tag-based Fair Queueing for Responsive and Unresponsive Flows.
- CSFQ : Core Stateless Fair Queueing [6]. We used the code of CSFQ they released for the ns simulator [14].
- FIFO : The standard FIFO router
- RED : The Random Early Discard router [15].
- DRR : The Deficit Round Robin router [5], which is an implementation of the weighted fair queueing discipline. DRR serves as a reference in terms of fairness for our simulations.

We reproduced most of the scenarios presented in [6] for better comparison. In all our simulations, the links have a capacity of 10 Mbps, and a propagation delay of 1ms. Our buffers length is 64 KBytes, and all packets are 1000 bytes long (ns default values). The simulation where run over a 10 seconds time interval.

To summarize the following results, we can say that for the two first sets of simulations (5.1, 5.2) CSFQ and TUF have comparable performance, and achieve a good degree of fairness slightly below that of DRR. In the third set of simulations, where we introduce bursty ON/OFF sources, we show that in certain conditions, TCP sources suffer from instability under the CSFQ scheme, but behaves significantly better with $\mathcal{T}$UF .

5.1 A single congested link

The first set of simulations is run with the topology shown on figure 8. The purpose of these simulations is to evaluate the fairness of the different mechanisms when a number of UDP and TCP connections share the same link.

In the first simulation, 32 UDP flows with varying aggressivity share the same 10Mbps bottleneck link. The arrival rate for the UDP flow number i is $(i+1)$ times the fair share rate, i.e. $(i+1)*10Mbps/32$. Figure 9a shows the throughput achieved by all the UDP flows. The RED and FIFO queueing disciplines do not ensure fairness, and the most aggressive flows (with the greatest flow IDs) achieve the best throughputs. On the other hand, DRR, CSFQ, and TUF propose comparable output rates to all the UDP flows.

The second simulation (figure 9b evaluates the impact of an ill-behaved UDP flow (Flow ID=0) on a set of 31 TCP connections. The UDP flow is a CBR source whose rate is the link's capacity, i.e. 10Mbps. Again, apart from RED and FIFO, that give most of the bandwidth to that flow, DRR, CSFQ and $\mathcal{T}$UF limit the rate of the UDP connection to its fair share, as all the other connections.

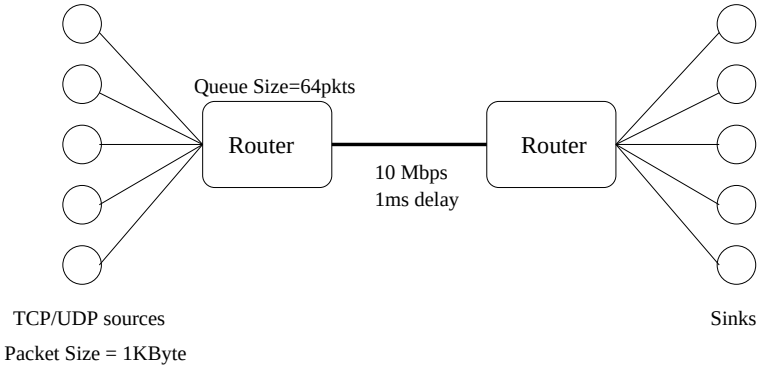

Figure 8: The single congested link topology

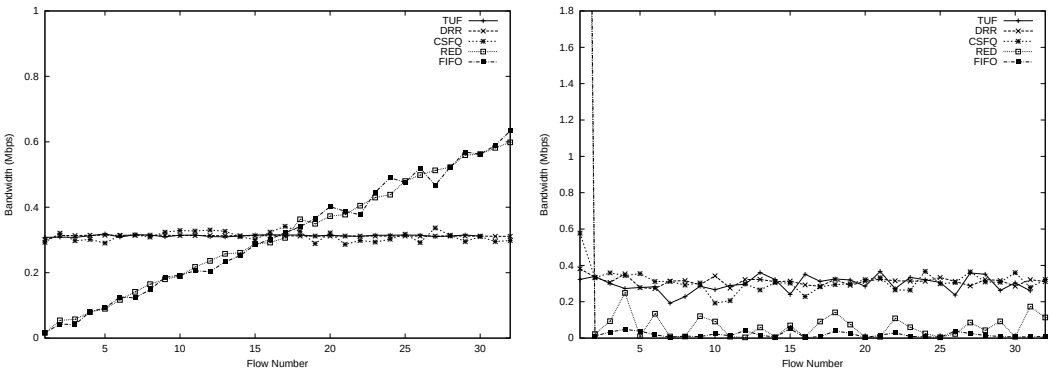

Figure 9: Throughputs for N=32 flows sharing the same 10 Mbps bottleneck link. (a)The arrival rate for UDP flow (i=0..32) is (i+1) times the fair share rate. (b)The UDP flow (ID=0) is sending at 10 Mbps shares the link with 31 other TCP flows.

In the third simulation, we show how a single TCP connection resists to the pressure of an increasing number of concurrent UDP connections. The UDP connections all send at twice the fair share rate (10 Mbps divided by the number of flows). Figure 10 shows the normalized bandwidth of the TCP flow, which is the ratio between the throughput achieved and the fair share rate, as a function of the number of flows. Note that, as explained in [6], the performances of DRR are significantly affected when the number of flows exceeds 22, because the space reserved in the buffer for the TCP connection is too small. CSFQ and $\mathcal{T}$UF performances are comparable : the TCP connection receives between 55% and 90% of the fair share rate for CSFQ and 70% and 100% for TUF.

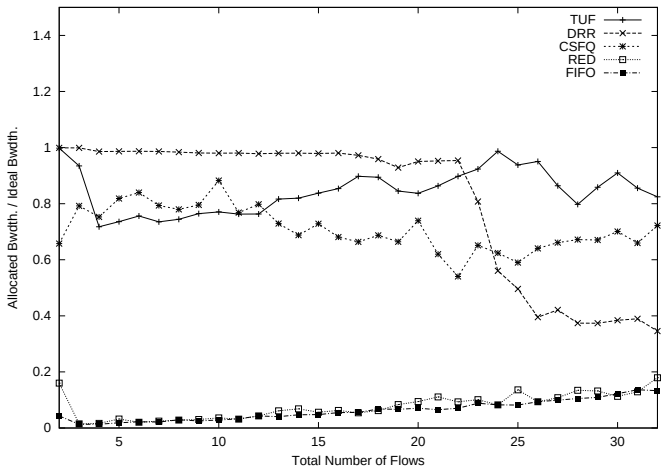

Figure 10: Normalized throughput of a single TCP flow sharing the link with N=1..32 other UDP flows. Each UDP flow sends at twice its fair share

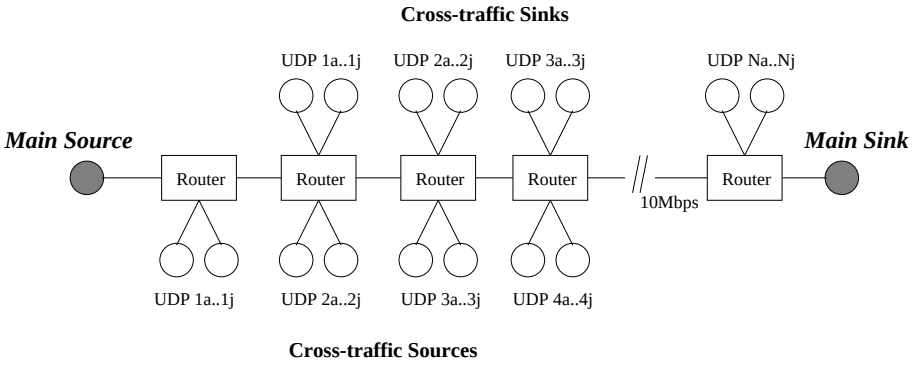

Figure 11: The multiple congested link topology

5.2 Multiple congested links

The second set of simulations is run with the topology shown on figure 11. The purpose of these simulations is to evaluate the robustness of these mechanism when the flows traverse more than one congested link. 10 CBR sources send at 2 Mbps on each of the congested links. All this cross traffic enters the path in one of the router and exits at the next. The main source is a TCP source (figure 12a) or a UDP source (figure 12b) sending at its fair share rate (909Kbps).

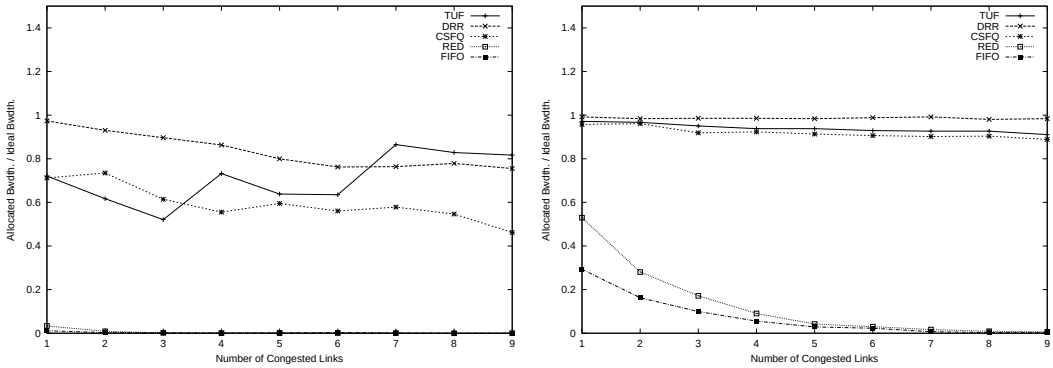

Figure 12: Normalized throughput of a TCP (a) or UDP (b) connection going through N=1..9 congested links. Cross traffic are CBR sources sending at twice the fair share rate.

The first simulation with a TCP connection shows once more that $\mathcal{T}$UF and CSFQ are comparable, and DRR performs slightly better. However, $\mathcal{T}$UF seems to perform better than CSFQ as the number of congested links grows. RED and FIFO routers do not enable the TCP connection to achieve a significant throughput. In the second simulation, the main UDP source is absolutely not affected by the cross traffic, and achieves perfect fairness in all three scenarios (DRR, CSFQ, and $\mathcal{T}$UF), whereas RED and FIFO are unable to maintain a significant throughput.

5.3 Bursty cross traffic

The third set of simulations is run on the same topology (figure 11). The CBR sources that formes the cross traffic are now replaced with ON/OFF sources. The burst (ON) and idle (OFF) time periods are both exponentially distributed with the same average chosen between 5ms and 1s. The cross traffic's average intensity is the same than for the previous set of simulations : the UDP ON/OFF sources send at 4 Mbps during the ON period.

As the ON/OFF time periods reach the critical 100ms, the performances of CSFQ are seriously affected (Figure 13). Figure 14 shows us the congestion window size and the losses for the TCP connection going through 3 congested links, both for CSFQ and $\mathcal{T}$UF . The average burst and idle times equal 200ms. We observe that burst of losses occur for CSFQ and not for $\mathcal{T}$UF . Indeed, when the cross traffic intensity increases, the fair share decreases and the TCP connection is using more bandwidth than it deserves. CSFQ reacts by dropping bursts of packets, to adjust the output rate to the fair share rate. This impacts dramatically on the performance of TCP. On the other hand, $\mathcal{T}$UF drops just one packet, and TCP reduces its throughput on its own once it detects the loss.

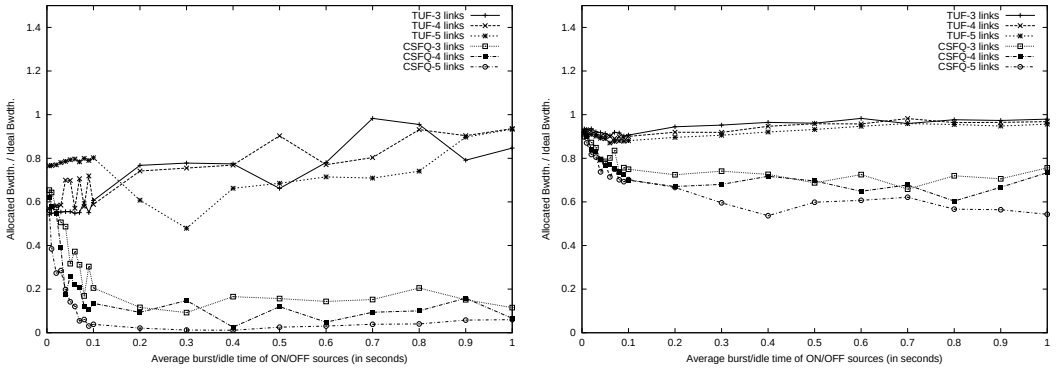

Figure 13: Normalized throughput of a UDP (a) or TCP (b) connection going through 3 or 5 congested links. Cross traffic are on/off sources whose burst and idle time vary between 5ms and 1s. The average rate of these sources is twice their fair share.

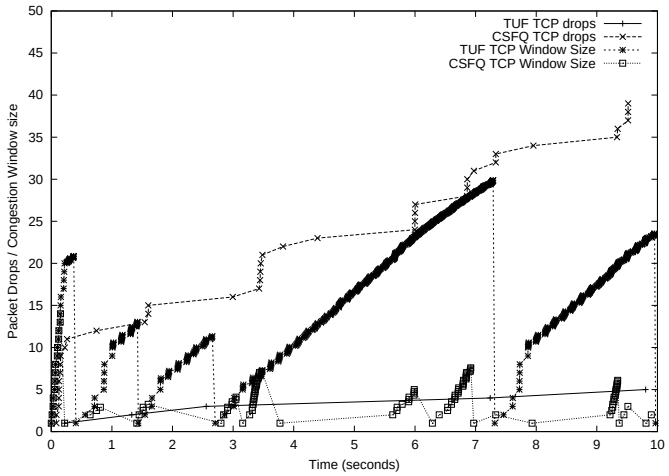

Figure 14: Congestion window size and losses for the TCP connection going through 3 links. Average burst and idle time equal 0.2s.

6 Conclusion

In this paper, we presented $\mathcal{T}$UF , a Tag-Based Fair Queueing for both responsive and unresponsive flows. This scheme enables us to do fair bandwidth sharing among flows of different types, such as TCP and greedy UDP flows, without requiring flow-state in the high-speed backbone routers. Packets are tagged at the edge of the network, or at the source, with a value that represent the minimum fair share rate a router must support to forward the packet. The queueing discipline in the $\mathcal{T}$UF router is very simple and consists in dropping the highest tag value when the queue is full. This behaves as a bufferless filter, that lets through packets with tag below a threshold K. We simulated $\mathcal{T}$UF , and it achieves approximately fair bandwidth sharing, as CSFQ or DRR. However, it provides better fairness than CSFQ in bursty environments, especially for TCP flows, since it adapts specifically to any form of responsive flow whose throughput can be determined as a function of the loss rate. We believe that, by its simplicity and efficiency, $\mathcal{T}$UF is an interesting approach to network oriented congestion control. We are currently working on a real implementation.

References

[1] Dong Lin and Robert Morris, "Dynamics of random early detection," in *Proceedings of ACM SIGCOMM'97*, Cannes, France, October 1997, vol. 27, pp. 127–136.

[2] Alan Demers, Srinivasan Keshav, and Scott Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proceeding of ACM SIGCOMM'89*, 1989, pp. 3–12.

[3] Jon C.R. Bennett and Hui Zhang, "Wf2q : Worst-case fair weighted fair queueing," in *Proceedings of IEEE Infocom'96*, San Francisco, CA, March 1996, pp. 120–128.

[4] Jon C. R. Bennett and Hui Zhang, "ierarchical packet fair queueing algorithms," in *Proceedings of ACM SIGCOMM'96*, October 1996, vol. 26, pp. 143–156.

[5] George Varghese, "Efficient fair queuing using deficit round robin," in *Proceedings of ACM SIGCOMM'95*, 1995, vol. 25, pp. 231–243.

[6] Ion Stoica, Scott Shenker, and Hui Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," in *Proceeding of ACM SIGCOMM'98*, 1998, vol. 28, pp. 118–130.

[7] Zhiruo Cao, Zheng Wang, and Ellen Zegura, "Rainbow fair queueing: Fair bandwidth sharing without per-flow state," To appear in Infocom 2000. http://www.cc.gatech.edu/people/home/zhiruo/zhiruo.html.

[8] "Reference removed for double blind reviewing," .

[9] Steven McCanne, Van Jacobson, and Martin Vetterli, "Receiver-driven layered multicast," in *Proceedings of ACM SIGCOMM'96*, 1996.

[10] Dimitri Bertsekas and Robert Gallager, *Data Networks*, chapter 6, pp. 524–529, Prentice-Hall, 1987.

[11] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM'98*, 1998.

[12] Ion Stoica and Hui Zhang, "Providing guaranteed services without per flow management," in *Proceedings of ACM SIGCOMM'99*, 1999.

[13] Steve McCanne and Sally Floyd, "Ucb/lbnl/vint network simulator (ns) version 2.1b5," http://www-mash.cs.berkeley.edu/ns/, 1999.

[14] Ion Stoica, "Csfq simulation scripts for ns-2," http://www.cs.cmu.edu/ istoica/csfq/, 1998.

[15] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399